

[0017] The features and advantages described in the specification and in this summary are not all inclusive and, in particular, many additional features and advantages will be apparent to one of ordinary skill in the art in view of the drawings, specification, and claims. Moreover, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the disclosed subject matter.

## II. Real-Time Messaging Platform Overview

[0018] FIG. 1 is a block diagram of a real-time messaging platform 100, according to one embodiment. The real-time messaging platform 100 (also referred to as a “messaging platform”) includes a frontend module 110, a pilot frontend module 122, a routing module 125, a graph module 130, a delivery module 135, a message repository 140, a connection graph repository 142, a stream repository 144, an account repository 146, and an unmanned aerial vehicle (UAV) interaction engine 150.

[0019] The messaging platform 100 allows account holders to create, publish, and view messages in a message stream visible to themselves and other subscribing accounts of the messaging platform 100. Account holders compose messages using a client software application running on a client computing device 105 (also referred to as a client 105 or a client device), such as a mobile phone, a tablet, a personal computer (laptop, desktop, or server), or a specialized appliance having communication capability. The client software application may include a web-based client, a Short Messaging Service (SMS) interface, an instant messaging interface, an email-based interface, or an API (application programming interface) function-based interface. The client computing devices 105 communicate with the messaging platform via a network. The network may communicate information through wired or wireless communication channels over a local-area network, a wide-area network such as the internet, or a combination thereof. The network may include multiple networks or sub-networks.

### II. A. Message Composition with a Real-Time Messaging Platform

[0020] Messages are containers for a variety of types of computer data representing content provided by the composer of the message. Types of data that may be stored in a message include text (e.g., a 140 character “Tweet”), graphics, video, computer code (e.g., uniform resource locators (URLs)), or other content. Messages can also include key phrases (e.g., symbols, such as hashtag “#”) that can aid in categorizing or contextualizing messages. Messages may also include additional metadata that may or may not be editable by the composing account holder, depending upon the implementation. Examples of message metadata include the time and date of authorship as well as the geographical location where the message was composed (e.g., the current physical location of the client 105).

[0021] The messages composed by one account holder may also reference other accounts. For example, a message may be composed in reply to another message composed by another account holder. Messages may also be repeats (or reposts) of a message composed by another account holder. Reposts may also be referred to as “retweets.” Generally, an account referenced in a message may both appear as visible content in the message (e.g., the name of the account), and may also appear as metadata in the message. As a result, the messaging plat-

form allows interaction with a referenced account in a message. For example, clients 105 may interact with account names that appear in their message stream to navigate to the message streams of those accounts. The messaging platform 100 allows messages to be private, such that a composed message will only appear in the message streams of the composing account and designated recipients’ accounts.

[0022] The frontend module 110 receives composed messages from the clients 105, interfaces with other internal components of the messaging platform 100, and distributes message streams to account holders. The frontend module 110 may provide a variety of interfaces for interacting with a number of different types of clients 105. For example, when an account holder uses a web-based client 105 to access the messaging platform 100 (e.g., through an Internet browser), a web interface module 114 in the front end module 110 can be used to provide the client 105 access. Similarly, when an account holder uses an API-type client 105 to access the messaging platform 100 (e.g., through an application native to an operating system of the client 105), an API interface module 112 can be used to provide the client 105 access.

[0023] The routing module 125 stores newly composed messages received through the frontend module 110 in a message repository 140. In addition to storing the content of a message, the routing module 125 also stores an identifier for each message. This way, the message can be included in a variety of different message streams without needing to store more than one copy of the message.

### II. B. Connections in a Real-Time Messaging Platform

[0024] The graph module 130 manages connections between account holders, thus determining which accounts receive which messages when transmitting message streams to clients 105. Generally, the messaging platform 100 uses unidirectional connections between accounts to allow account holders to subscribe to the message streams of other account holders. By using unidirectional connections, the messaging platform allows an account holder to receive the message stream of another account, without necessarily implying any sort of reciprocal relationship the other way. For example, the messaging platform 100 allows account holder A to subscribe to the message stream of account holder B, and consequently account holder A is provided and can view the messages authored by account holder B. However, this unidirectional connection of A subscribing to B does not imply that account holder B can view the messages authored by account holder A. This could be the case if account holder B subscribed to the message stream of account holder A; however, this would require the establishment of another unidirectional connection. In one embodiment, an account holder who establishes a unidirectional connection to receive another account holder’s message stream is referred to as a “follower”, and the act of creating the unidirectional connection is referred to as “following” another account holder. The graph module 130 receives requests to create and delete unidirectional connections between account holders through the frontend module 110. These connections are stored for later use in the connection graph repository 142 as part of a unidirectional connection graph. Each connection in the connection graph repository 142 references an account in the account repository 146.

[0025] In the same or a different embodiment, the graph module 130 manages connections between account holders using bidirectional connections between account holders.